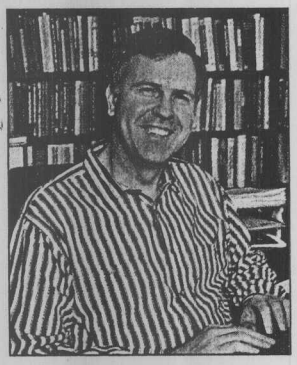
EDN | COLUMNIST



DAVID BRUBAKER,
FUZZY-LOGIC
CONTRIBUTING EDITOR

Design and simulate your own fuzzy setpoint controller

BBS

I enjoy backpacking—a lot. Colin Fletcher is a long-time expert in the field. His first backpacking how-to book appeared in 1968, and in its preface, he questions the value of "how-to" books in general. He wonders whether a novice would actually stand to gain more by just doing—and learning from his or her own errors—rather than following the advice of an expert.

I wonder the same thing now, but in a fuzzy context. Most neophytes benefit from learning from someone who has already "been down the road." In addition, there is some value, once you have been down that road yourself, in seeing how others work out the same, or similar, tasks. However, I fear that well-intentioned, fuzzy-logic fans might become too comfortable as listeners—rather than doers—and, thus, be uncomfortable with the prospect of struggling with their own ideas. Knowledge grows from being taught, but wisdom is gained through experience.

If you feel at least a part of your future may be fuzzy, I strongly recommend active participation. Buy a tool and start playing with it. You have lots of tools to choose from, but, for a number of reasons, I recommend CubiCalc version 2 from HyperLogic (Escondido, CA (619) 746-2765)). It is a strong fuzzy development tool with a powerful simulation language that allows you to develop rather complex fuzzy models. In its basic form, CubiCalc version 2 costs $495.

Now, for the example (the project file for the following CubiCalc design example is available via the *EDN* BBS). In this column (and the next several) we'll work together to design and simulate a simple fuzzy setpoint controller. We'll also consider the steps necessary to implement the simulation in a microcontroller-based system. The purpose of the controller is to charge a capacitor to a specified value through a series inductor and a very small series resistor (**Fig 1**). The capacitor is 100 μF, the inductor, 10 mH, and the resistor, 1Ω. The circuit resonant frequency is nearly 160 Hz, with a period of 6.25 msec.

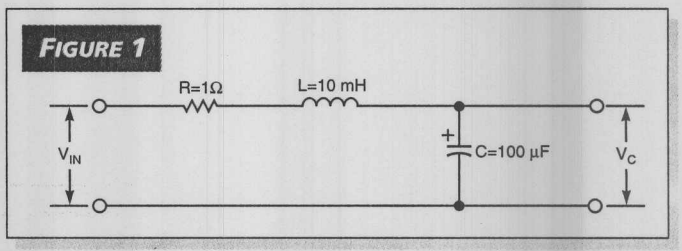
The purpose of the controller (**Fig 2**) is to apply the appropriate V_{IN} to drive V_C through a 5V step. The controller need only respond to a step; ramping or tracking arbitrary waveforms is not a requirement. Also, the amplitude of the step is constant. Response requirements are

- 0 to 90% risetime ≤1 msec
- Overshoot ≤5%
- Settling to ±1% in ≤2 msec.

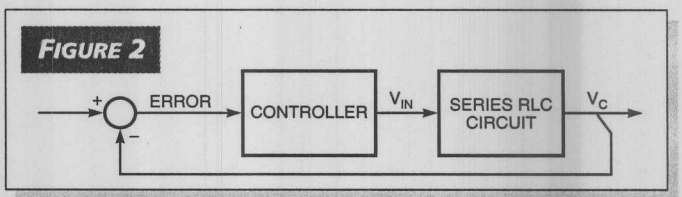
There are two additional constraints. First, V_{IN} is supply-limited to ±15V. Second, although the required step size is constant, the value of C can change from 50 to 200 μF.

Let's start with a series of design decisions.

Controller type: This is a second-order, primarily linear system—the ±15V drive limitation is the only nonlinear component. For constant valued C and constant step size, a linear PID controller does an excellent job, even with the nonlinearity. A linear PID controller is also easy to design. However,



The controlled system: The goal, here, is to charge the capacitor through the resistor-inductor series combination by applying a voltage, V_{IN}.



The closed-loop system: Based on the value of an error, the difference between the desired and actual values of VC, the controller applies an appropriate voltage, V_{IN}, to the RLC circuit.

PHOTOGRAPHY / LIANE ENKELIS

The listings mentioned in this article are available on the EDN Readers' BBS. Phone (617) 558-4241 with modem settings 1200/2400 8,N,1 (9600 baud=(617) 558-4580). From the Main System Menu enter *ss/freeware*. Then from the /freeware SIG menu enter *rk0195bru*.

the linear system does worse over the range of capacitor values. Therefore, we will use a fuzzy controller.

Although this is a legitimate design choice (ie, linear vs fuzzy), for our discussion the point is moot—this *is* a fuzzy column.

Input selection: Selecting inputs to a fuzzy system is critical. That a fuzzy design is poor may have nothing to do with the use of fuzzy logic and everything to do with choosing poor inputs. Independent of system type, if desired system operation cannot be achieved in response to the selected inputs, the system won't operate properly. It doesn't matter what is in the middle; garbage on the inputs results in garbage at the outputs.

Unfortunately, even though sensitivity theory is useful in identifying whether a potential input is of value, I know of no systematic method for actually selecting inputs. Experience is invaluable.

For this example, we'll use the traditional proportional (P), integral (I), and derivative (D) components of measured error, ϵ, the instantaneous difference between desired and actual capacitor-voltage values. We'll use these inputs in a manner similar to a traditional PID system, with the P term used to reduce absolute error, the D term to control overshoot, and the I term to eliminate offset error near the setpoint.

Calculating I and D: A PID controller implemented with analog electronics has its integrator and differentiator built in. Although this is an option—a dual op amp and a few resistors and capacitors would do the trick—we'll calculate the integral and derivative of the error signal digitally. (See **Fig 3** for an explanation of this technique.) For the derivative, the slope of the signal between sample points is calculated. For the integral, the area under this same segment is calculated and added to the previous integral value.

There is some error associated with these straight line approximations, but oversampling lessens this error. The resonant frequency of the LC combination is approximately 160 Hz, or a period of 6.25 msec. The sampling interval is 50 μsec.

Normalized fuzzy values: I prefer working with normalized fuzzy values. That is, in creating membership functions, I size inputs and outputs to range from –1 to +1 if they are bipolar and from 0 to +1 if they are unipolar. Actual inputs and outputs are scaled appropriately.

I do this for a number of reasons. First, you can construct membership functions and rules at a level of abstraction above the actual system. For example, in the controller we are designing, we do not yet need to worry whether the P term (proportional error) will range between ±5, ±6, or ±10V. Rather, we limit its range to between ±1 and adjust its scaling gain accordingly. The same applies for the D and I input terms—and for the outputs.

Setting the actual scaling gains is then typically performed as part of the simulation step (if one exists), or test step (if no simulator is used). Initial input gains are set to just prevent saturation of the inputs and are changed only if inputs are later seen to saturate. Output gains are set to achieve a desired loop gain and are subsequently changed as part of the system tuning process, along with membership function and rule changes.

Number of rulebases: The standard approach for a three-input fuzzy system is to use a single, three-input rulebase. Doing so demonstrates what is currently called "rule explosion."

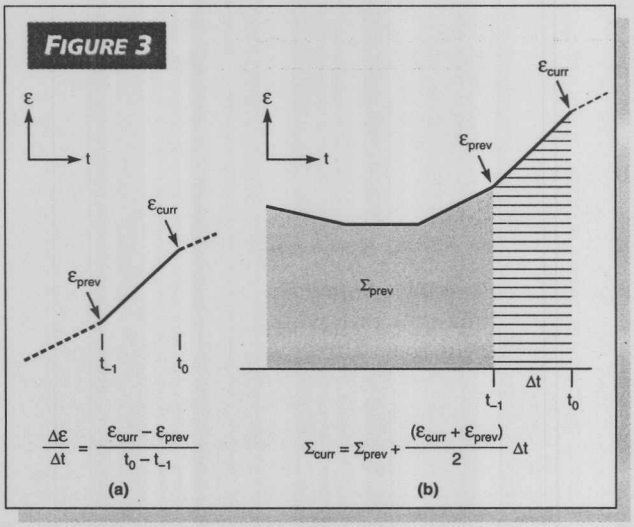
Let's represent each input as seven membership functions (with names, for example, NL=negative large, PS=positive small, and so forth). Five or seven membership functions for each input are fairly common. For a single-input system, you'll have seven rules, one rule for each input. Here is an example:

IF Input is NL THEN Output is PS;

For a two-input system, there are 7×7=49 possible input combinations and, therefore, if we fully populate the rulebase, 49 rules, which are now of the form:

IF Input_1 is NL AND Input_2 is ZR
THEN Output is PS;

FIGURE 3



Calculating the D and I terms: Use a straight line approximation for the derivative and a trapezoidal approximation for the integral.

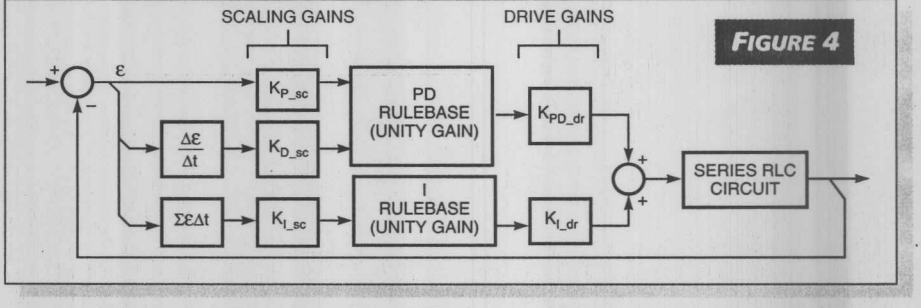


FIGURE 4

A fuzzy PD+I controller uses two rulebases, one for the P (proportional) and D (derivative) inputs, and a second for the I (integral) input. The amplified outputs of the two rulebases are summed and applied as the input to the controlled system. The input domains of all three terms (P, D, and, I) are normalized to ±1 units, thereby requiring both scaling gains on the inputs and drive gains on the outputs.

Similarly, for three inputs and a fully populated rulebase, there are 7×7×7=343 rules, each of the form:

IF Input_1 is NL AND Input_2 is ZR
AND Input_3 is NS
THEN Output is PS;

I, however, would rather not determine, generate, test, and tune 343 rules.

Three general approaches for getting around rule explosion are available. First, trainable systems, such as neural nets or genetic algorithms, are increasingly used to design fuzzy membership functions and rules. Be very certain of the training data. This is an excellent opportunity for garbage-in/gar-bage-out.

Second, some researchers are looking toward reducing the number of input membership functions (and thus the number of rules) based on the shape of the response function. This looks promising—when you know the shape of the response function.

Finally, complexity-reduction techniques currently used in simplifying large systems may also be appropriate. If the system is amenable to a "divide-and-conquer" stratagem, use it.

We'll use this latter approach. Knowing the operation of a linear PID controller, where individual gains are applied to the P, I, and D terms with the results then summed, I decided to see if the I term could be handled separately. This would result in a system with 49 rules used in the PD rulebase and 20 in the I rulebase (in two months we'll see why so many rules are used for the I term). The resulting structure is what I call a fuzzy PD+I controller. **Fig 4** shows the block diagram of the system incorporating fuzzy PD+I control.

We are running out of space. Next month we'll discuss the next step in the design process, which is how to flesh out our generalized design into membership functions, rules, and gains.

One final comment: We are designing a setpoint controller that controls an output parameter to a commanded value. I have chosen this to demonstrate how to design a simple fuzzy system. Although many setpoint controllers have already been developed, especially in Japanese consumer products, most fuzzy practitioners are quick to point out that fuzzy logic applied to control is more effective in task control. An example of task control Professor Zadeh often suggests is parking a car. The theory and application of setpoint control is quite well established and, although fuzzy systems often do a good job, other control methods also do so. Implementing task control with traditional methods tends to be more difficult, because we are forced to work with crisp threshold.

As always, your comments and feedback are welcome. If you E-mail, please include your postal address, too, in case my server can't find yours.

David Brubaker is a consultant in fuzzy-system design. You can reach him at Huntington Advanced Technology, 883 Santa Cruz Ave, Suite 31, Menlo Park, CA 94025-4608 or on the Internet at: brubaker@cup.portal.com.